the 4th generation

Reprinted from
DATAMATION.
January, 1967

the 4th generation

FOURTH-GENERATION SOFTWARE, by *Ascher Opler.* *Using large-scale integration logic specialized by microprogramming, the next generation of hardware may have expanded capabilities due to "firmware"—resident microprograms in the computer's control memory.*

FOURTH GENERATION HARDWARE, by *G. M. and L. D. Amdahl.* *Characterized by batch fabrication and increased economic imbalance between processors and peripherals, computer families to come will be faster and cheaper than ever.*

COMPUTING IN THE 1970'S, by *R. L. Patrick.* *Sent on a visit into the future, the author summarizes likely developments in the next few years and goes to the 1973 FJCC.*

THE NEXT GENERATION. *What will it feature and when will it be here? A roundup of comments from hardware and software specialists and from users with a variety of requirements.*

FOURTH–GENERATION SOFTWARE

by ASCHER OPLER

On the assumption that the next generation of information processing systems should try to correct the shortcomings of its predecessors, let's preface our sneak preview of the fourth generation with a quick review of the major general failures of third-generation software.

Third generation dissatisfaction centers on the disappointing price/performance ratio of the hardware/software combination (from now on we'd better say "software/hardware"!) and conversion difficulties. The disappointing performance is generally attributed to software.

At one time we used hardware price/performance ratio as a useful measure. When software became an indispensable part of hardware, we measured only the performance of the combination. Since the separate pricing of software is upon us (e.g., Sigma 7 COBOL), it is likely that we'll soon be measuring software price/performance ratio. With this in mind, we can look at the price/performance ratio of the third-generation combination and observe that the *hardware* ratio is excellent but the *combined* ratio is generally disappointing.

The conversion picture is generally bleak. Those who used FORTRAN and COBOL heavily with second-generation equipment and those who can emulate their second-generation computer on their third come off well compared to those who didn't use the two languages or who can't emulate.

projected fourth-generation computers

This article is based on one man's guess about the next generation. The guess is that fourth-generation hardware will use large-scale integration (LSI) logic specialized by microprogramming. Integrated logic chips will be needed to obtain the required fast circuitry and the microprogrammed specialization will be needed to obtain the required software and user interfaces.

The elements of one of these computers might have these characteristics:

circuitry (LSI)	5 nanoseconds
micromemory access	10 nanoseconds (read time)
main memory access	400 nanoseconds

In the most radical departure from present architecture, the computer would have *no order set* and *no data structure*. The computer would be specialized for the various roles it is to play by *replaceable microprograms*.

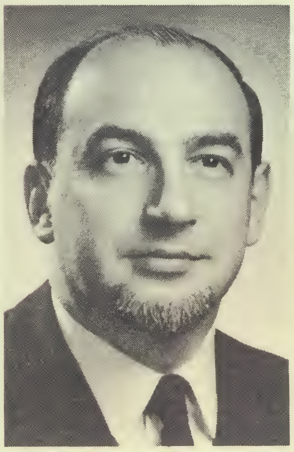
Third-generation microprogrammed computers are delivered with a pre-designed, pre-installed microprogram of the read-only type. And in most members of the product line, the option of an additional read-only memory carrying a *second* microprogram set is available. This second set serves to provide full or partial compatibility with the order set of an older computer.

In fourth-generation computers, *many* microprograms will be available from the manufacturer. Software and user specialists will also prepare and use their own. This should throw the whole field wide open.

To better understand the nature of microprogramming a no-order-set/no-data-structure computer, I believe it worthwhile to introduce a new word into our vocabulary: *firmware*. I use this term to designate microprograms resident in the computer's control memory, which specializes the logical design for a special purpose, e.g., the emulation of another computer. I project a tremendous expansion of firmware—obviously at the expense of hardware but also at the expense of software.

Microprogramming has, of course, been known since the early days of computing. The principle is simple—build a small micro-processor and drive it with a stored microprogram.

Take floating-point addition as an example. In the first generation, it was performed with a normally-programmed subroutine. In the second generation, it was implemented in circuitry. In third-generation microprogrammed computers, the floating-point addition is performed using a microprogram stored in read-only memory.



A noted software specialist, Mr. Opler is executive director of Computer Usage Education, Inc., in New York City. He was formerly director of programming systems for the parent company, Computer Usage Co.

One major factor making this possible is the ratio between main memory access time and circuitry speed. With integrated circuits (IC) and now LSI, times have *plunged* to 1-5 nanoseconds while the access time of main memory has *slowly* declined. This allows 30 to 70 micro-steps to be performed by circuitry between each main memory fetch or store. These 30 to 70 steps permit the implementation of third-generation order sets.

The memories are called read-only memories but, of course, the contents had to be written once in order to "load" the memory. Today this is done as part of the computer fabrication process. For the fourth generation, extension to slow-write fast-read (SW/FR) memory is anticipated.

The interface between "normal" programs and microprograms comes via the operation code. In effect, when each instruction is decoded, the operation code calls upon a specific microprogrammed "routine" stored in the micromemory.

To visualize the preparation of firmware, consider a special keypunch not available to software and user programmers. This punches 67-column cards (4 x 8 inches) with triangular holes (try that on your keypunch and card reader!). A special card reader loads decks of these 67-column cards into the fourth-generation SW/FR memory. This card reader probably will be locked so that only firmware specialists have access to it (we hope)!

what firmware can do now

Even in the third generation, firmware is expanding the capabilities of hardware and software. Its effect on reducing the cost of hardware is central to third-generation economics. Unprecedented production rates with reduced requirements for system checkout and delivery are due to both integrated circuitry and to microprogramming.

Specialized versions of standard product line computers have been prepared primarily by altering the microprogram in the read-only memory. The most spectacular success has been achieved with all-hardware compatibility via a special microprogram. The success of the IBM System/360 Model 30 in running IBM 1401 programs attests to this fact.

Emulation, a combination of software and hardware, operates by using two different microprogrammed features. Basically, an emulator is an interpretive simulator made much more efficient by using a microprogram to perform the basic interpreting loop—fetch an instruction, decode the address, access the contents of the address and perform the indicated operation. In addition, the op-codes that are most time- (or space-) consuming to simulate are directly executed by microprogramming those orders into the micromemory. Emulation is, of course, required when the micromemory is too small to contain the entire order set of the second-generation computer.

An interesting example of the power of extending existing capability by microprogramming is the evaluate (EVAL) instruction added to the IBM System/360 Model 50 delivered to Allen-Babcock Computing Inc. This command is reported to evaluate a PL/I expression in standard form directly in hardware. Other added facilities include *floating decimal* instructions.

firmware in the fourth generation

Assuming the availability and accessibility of adequate SW/FR memories in the next generation, the entire *hardware/software* interface problems disappear only to be replaced by the more complex *hardware/firmware/software* interfaces. First, there is the hierarchy of components and corresponding responsibility.

Component	Responsibility	Organization
1. Circuitry	Hardware Designer	Manufacturer
2. Micromemory	Firmware Specialist	Manufacturer*
3. Control Programs	Software Specialist	Manufacturer*
4. Processors, Utilities, etc.	Software Specialist	Manufacturer*
5. Application programs	Application Specialist	User

*and/or software (firmware) producing organizations.

Secondly, firmware will assume a dominant role in structuring the computer. Manufacturers could (potentially) supply firmware decks allowing their fourth-generation computer to execute the order set of most popular third-generation computers—their own and competitors'. There could be a standard fourth-generation industry-wide order set and data structure.

But an even more significant impact would be felt by software. There is a general consensus that present software is too large, too complex and too slow. It is in the tradeoffs between software and firmware that most price/performance improvement should be obtained.

Third-generation computers require control programs to resolve and handle interrupts, to control multiprogramming and input/output dispatching, and to provide useful services such as resouce allocation and protection, etc. The demands placed on control programs by the hardware

and by the user have forced the current programs to swell to enormous size with concomitant reduction in performance.

For fourth-generation computers, the answer will lie in using firmware for major portions of the control program functions and in using special features built into the firmware to facilitate other control program functions.

For instance, one problem that contemporary software must solve is to provide flexibility so that any particular installed configuration can be used with "general purpose" control programs. Even though "system generation" is used to specialize a generalized control program for a given configuration, the resultant code often contains lengthy subroutines to handle such functions as resolving device error signals, providing special user error recovery or interrupting the processing routine. The point is this—to provide modularity and flexibility, manufacturers are currently forced to delegate obvious hardware functions to software since this is the only means by which the user of the computer can specialize it for his use.

In fourth-generation equipment using SW/FR micromemory, microprograms can be prepared by the firmware specialist—manufacturer, programming company or the user—to carry out the specific interrupt and input/output control function specified by the user. This alone will go far to simplify control programs.

Further simplification can be obtained by making the data structure and order set work for, not against, the implementors of control programs. The basic implementation involves techniques of queue management, control block handling, table reference, internal sorting, pointer handling, etc.

Since microprogramming permits extensive data structuring for control program implementors, it will permit the addition of instructions to enqueue, dequeue control blocks build search and sort tables of specified structure, etc. These new commands should prove a boon to expediting the running of supervisory programs.

Time-sharing and multiprogramming both require very fast switching among programs. Many fourth-generation computer programs (processors and utility programs) will be written in the form of re-entrant code. The new hardware must be able to preserve the status of each user's computation and, when re-entrant code is used, to preserve blocks containing variables and other parameters. Currently much of this preserve/restore function is performed by software. Delegating this function to firmware should reduce the milliseconds performance time to microseconds.

compilers and firmware

Last year, a paper by Melbourne and Pugmire[1] described the design of a small computer for *directly executing* FORTRAN *statements*. The machine, which was simulated but not built, was "controlled by a microprogram held in a fixed store." Marketing men can determine the reception of a FORTRAN-only computer—but it is certainly one reasonable approach for the fourth generation. With SW/FR firmware, the FORTRAN microprogram can be read into micromemory—instead of the order set of a computer.

Short of a computer which executes programming language directly, firmware can add many features to fourth-generation computers to facilitate compilation and execution. Special instructions corresponding to the POP (programmed operators) of Digitek compilers can be microprogrammed to facilitate compilation. Facilities like the "hardware algorithms"[2] of System/360 Model 91 can operate on object code to permute computational order and even eliminate redundancies. Direct execution of statements in Polish string notation—a technique first used in the KDF9 and B5000—can add to the power of new computers which handle programming languages with unprecedented speed and economy.

It is quite likely that the bulk of programs run on fourth-generation computers will be written using standard programming languages. These may be run in some multiprogrammed (or time-shared) manner with other programs on a large computer run as a "powerhouse utility" serving many users — or on smaller "private" computers dedicated to a single application use (e.g., message switching) or the needs of a single isolated user (e.g., small engineering laboratory). The latter may very well prefer to use the FORTRAN—only (PL/I only?) computer while the user of the dedicated equipment may find it to his advantage to start from scratch and write the central portion of his program in firmware using normal core (or thin film) storage only for bulk tables and transient storage.

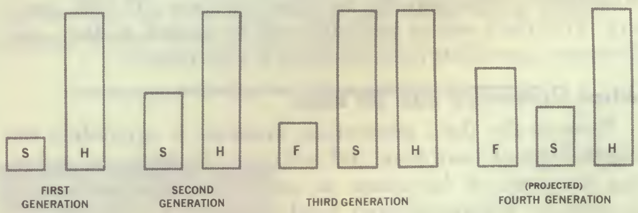
Depending on the ease of preparing, debugging and loading firmware, the whole application area might be radically altered. But it will probably be worthwhile to prepare firmware only for real-time, dedicated applications where every microsecond (nanosecond?) counts!

At the present time, there is a limited supply of microprogrammers and a *relatively* small demand for such services. The only firmware under preparation is in the shops of manufacturers and is performed under careful supervision of hardware designers. Preparing microprograms is considerably different from writing programs. It requires a more detailed knowledge of the function of circuits and registers.

Automatic preparation of microprograms starting with some "higher" level language—like FLOATING ADD A,B—has not been achieved. However, as firmware becomes more important, increased effort to facilitate its preparation will certainly be made. Indeed, if some of the many roles that firmware is to play in the fourth generation are to be realized, a new generation of specialists must be trained and effective tools provided for their use.

perspective

The following chart shows the relative effort (compared to the manpower involved in the production of the hardware of a computer) expended by manufacturers for different computer generations.



With increased demand for low-cost, high-performance total systems, the emphasis continually shifts to means of buttressing hardware to make it easier and more efficient to use. The third generation has seen the peak of the effort in software. No doubt the fourth generation will require equivalent effort — but placing much of what was software into the firmware area should go far to prevent some of our current difficulties. ■

[1]Melboune, A. J. and Pugmire, J. M.: "A Small Computer for the Direct Processing of Fortran Statements" *The Computer Journal*, Vol., 8, pp. 24-28 (1965).

[2]Chen, T. C., "The Overlap Design of System/360 Model 92 CPU" AFIPS Proceedings, Vol. 26, Part II, pp. 73-80, (1964).

FOURTH-GENERATION HARDWARE

by G. M. and L. D. AMDAHL

". . . visiting the iniquity of the fathers upon the children to the third and fourth generation . . ." —Exodus 20:5

It has been recently discovered that a third generation of computer technology has been entered. With the clarity of this observation we need no longer obliquely ask what the trend is in computer hardware or software, but we can ask directly and forcefully, "What will the next generation bring?" Accepting the notion that there will be a fourth generation of computers (by the principle of binary powers, eight years after the third) the following observations and extrapolations are put forward.

The third generation of computer technology can be characterized as one of unification. An attempt has been made to unify in one computer structure the effective capability to perform commercial, scientific and real-time tasks. Time-sharing is being elaborately analyzed, with conversational mode systems in early use. Monolithic integrated circuits are coming into widespread computer use, but with their greatest potential yet to be realized.

To achieve software interchangeability, the concept of upward compatibility has been extended by some to include downward compatibility, giving rise to families of computers differing only in speed and price. Manufacturers such as IBM found that this could be accomplished only by drastic means—design of a totally new line of computers having basic differences from predecessor equipment. The transition to the third generation has therefore been arduous, with techniques such as microprogramming and emulation being employed and with complete new software being required.

fourth-generation hardware: batch fabrication

Computer hardware of the fourth generation is anticipated to be most strongly characterized by batch fabrication. In areas of logic circuits and high-speed memory, complex arrays of switching circuits on single silicon chips will be batch fabricated. This is referred to as large scale integration, or more simply, LSI. It should be noted that batch fabrication does not necessarily mean the concurrent fabrication of identical items; rather, it means the concurrent fabrication of many different items passed through identical process steps. This permits economically produced silicon chips to logically differ. While large parallel computers (such as the proposed Illiac IV) could have many identical chips, small computers would be denied this luxury. Therefore the economic production of many different kinds of chips would benefit the manufacturer of the small computer.

Integrated circuits offer benefits of batch fabrication both in the area of logic circuits and memory. The benefits of LSI for logic circuits will be reduced cost, greater density and greater speed. The limitations of LSI in this use arise in interconnections, heat dissipation and chip complexity. To use LSI, a problem of the first magnitude will have to be solved: the complete automation of chip design. This problem is vastly complicated if yield must be enhanced by adjustment of the interconnection pattern of cellular circuits as a function of test results.

LSI memories can be very high speed and can permit multiple-access use. Because select and sense circuits are fabricated by the same process steps as those used for memory cells, high performance memories are expected to be relatively inexpensive. An economic property of considerable advantage for small memories is the relatively linear cost per bit as a function of size exhibited by LSI memories. But these speed and cost advantages will not apply to large LSI memories. Another factor to be considered by systems designers is the volatile nature of the LSI storage cell, permitting loss of information stored when power is removed.

Other batch fabrication techniques for memories will undoubtedly emerge in each of the areas of wired ferrite arrays, woven plated wire arrays and thin films. Any of these will be suitable for large quantities of lower-speed random-access memories and would not have the volatile characteristics of the LSI memory.

Mass storage still appears to be dependent on electromechanical devices for lowest cost implementation. To a large extent this is due to the fact that batch fabrica-

Gene Amdahl was project engineer and chief designer for the IBM 704, and initial planner for the IBM 709 and 7030. Later managing data processing engineering at Aeronutronic, he rejoined IBM in 1960 to serve as manager of architecture for the System/360. Now an IBM Fellow working on advanced computing systems in Sunnyvale, Calif., he received a Ph.D. in theoretical physics from the Univ. of Wisconsin.

tion is always easier to achieve when detailed structure is unnecessary, and the fabrication of the surface of a disc or drum is indeed batch fabrication of an enormous number of storage cells. However, the feverish activity in the development of static (nonrotating) mass storage will surely result in its use in fourth-generation equipment, particularly for systems requiring rapid information retrieval.

architecture: an attempt to balance

The economic consequences of batch fabrication are expected to have the greatest effect on fourth-generation architecture. Even in the third generation, processor costs are relatively low compared to other system cost factors such as I/O channels, peripherals, marketing and software. In the fourth generation this imbalance will tend to increase, with the system architect offsetting it by greater instruction capability, more processing overlap and parallelism, and additional hardware features for multiprogramming and multiprocessing.

The use of the high-speed LSI memory will tend to impose an additional level in the memory hierarchy. This will emphasize the need for semi-automatic control of memory, perhaps along the lines of paging techniques. We would expect that considerable emphasis will have been applied to this area due to difficulties incurred in third-generation systems. The problem here is to make the memory appear to be a very large and homogenous virtual memory (techniques which permit viewing main memory as unrestricted by actual main memory limitations), yet without imposing virtual response and virtual solution times.

Basically the computer can only deal with information residing in real memory. It must in some manner be provided with instructions which, during its operation, can cause it to control the transmission of information between hierarchy levels with as much preplanned structure as is possible. In circumstances where preplanning is inadequate, multiprogramming must be able to fill the voids.

Failure to provide any preplanning would raise the level of multiprogramming, requiring memory capacity that would be considered excessive even in the era of batch fabrication. The solution time for individual large problems may become excessively long in a multiprogrammed environment, making the system unsuitable for this purpose. Algorithmic control, suited to access characteristics of data sets, will undoubtedly become important. The nature of the control would be governed by programmer specification or by recognition of historical referencing properties as determined by the hardware.

Another area of architecture is the employment of microprogramming to an extent that compiler languages might be optimized. In a companion article, Ascher Opler predicts a fuller utilization of this technique in the fourth generation and the emergence of *firmware* specialists.

The question of equipment dependability will come under careful scrutiny in the fourth generation despite the fact that LSI will offer high reliability. The reason for this scrutiny will be the customer's concern for nearly absolute computer up-time in time-sharing environments. While in the past he was willing to accept turnaround delays due to the computer being down, he will not accept inoperative on-line terminals. Partial redundancy, multiple processors and in-line diagnostics should result from this emphasis. With LSI, redundancy can more economically be achieved than with discrete components.

software: more freedom

The role of software has been changing from generation to generation. From an initial start of freeing man from dealing directly with the computer by the provision of input-output utility programs, the role has progressed from having to deal in absolute terms by means of symbolic program assembly, from having to deal in basic terms by means of macro assembly and compilation, and finally from having to deal with space-time boundaries by means of data-management programs and virtual memory. One of the current activities is aimed at freeing man from space and time separation from the computer by time-sharing.

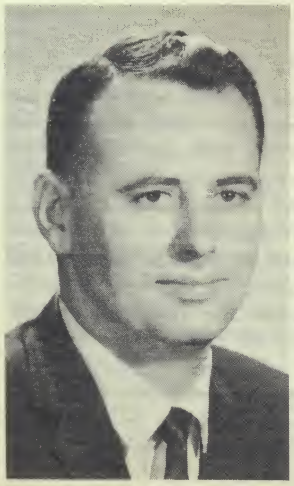
One can speculate as to those freedoms which will be sought for the next generation. A number of likely candidates come to mind, many of which have been started and have varying degrees of progress. Some fairly basic ones are (1) freedom from redundant effort by means of common libraries of programs and data sets in the information utility, (2) freedom from limited forms of man-machine communication by means of better techniques for the identification, extraction and display of meaningful information, (3) freedom from necessity to recognize the particular machine type being used by software standardization, and (4) freedom from painful recovery on machine malfunction by automatic detection and by maintenance of user transaction journals.

It is interesting to note that these freedoms contribute to a layer of basic concepts. These concepts relate the more abstract notions of computing to the detailed dynamic activity of the hardware computer. As more freedoms are added which release the user from primitive representation of his problem, he is further insulated from the hardware. This intervening software structure will develop into a capability for self-generation of programs for specific computational actions. Such programs will enjoy an existence so brief that they deserve a unique name, for which *bubbleware* is advanced.

One can deduce that significant overhead will be added in the execution of hardware functions as the user gains freedom in detailed program declaration. Despite this, the hope for fuller realization of human creativity will give continued impetus for this trend to greater abstraction.

beyond the fourth

It may be inappropriate to speculate beyond the fourth generation, but it appears that the logical conclusion to be drawn is that the ultimate goal must be freedom from having to deal with the computer at all. When this level of development has been reached, creative intellect will have burst its final shackles, free to wing to new Olympian heights. ■



Lowell Amdahl is president of COMPATA, Inc., a Tarzana, Calif., hardware consulting firm. A technical adviser to DATAMATION, and a participant in the design of several computers, he has a B.S. in engineering and physics from S. Dakota State and an MS in physics from Kansas Univ.

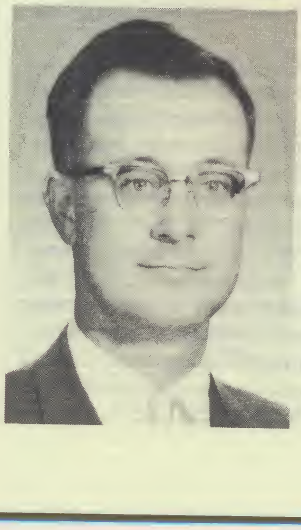
COMPUTING IN THE 1970'S

by R. L. PATRICK

(As our able editorial adviser pushed out into the future, he recalled some of the developments taking place since 1966 while enroute to a hypothetical 1973 FJCC. . . .)

By 1968 the shock of IBM's System/360 had largely subsided. It had been an expensive experience for manufacturer and user alike. Multi-programming generally worked fairly well, but the software overhead was greater than expected. System generation was no longer the chore it once was, and software changes no longer came with disconcerting frequency. The users, while not quite euphoric, had licked their wounds, completed their conversions, and were generally happy with the product and the results they were getting. Some big shops had finally embraced PL/I totally and were working the bugs out of using a single "universal" language. Programmers were genuinely interchangeable, since they all enjoyed the same language and operating system. This, however, did not make them all equal since differences in technical background, experience, and ability were still very apparent.

In late '68 and early '69 the users discovered the benefits of modularity. The manufacturers have been making systems more or less modular for years. When S/360 and its counterparts hit, the manufacturers reaped many benefits immediately from the building block approach. In 1968 the user discovered that this could help him too. As a company's contracts fluctuated, the computing manager adjusted his configuration to match his load. The manufacturers responded by building popular equipment units for regional inventories to allow reconfiguring with ease. With the advent of multi-programming, the trimming could satisfactorily take place by adjusting the number of I/O channels and devices to allow fewer tasks to operate simul-



Mr. Patrick is an established free-lance computer consultant based in Northridge, Calif., whose clients are involved in research, applications, operations and military developments. Holder of a CDP, he is also an editorial adviser and a frequent contributor to DATAMATION.

taneously. Thus, throughput and cost could be moderately changed without changing the cycle time of the CPU or affecting the unit cost for a specific application.

The shake-out of primary manufacturers continued and we lost several. Those that remain are stronger and better able to compete. Two of the also-rans have firmly established themselves as suppliers of consoles and I/O devices. They discovered that IBM's standard hardware interface* allowed them to design to one electrical specification and penetrate the IBM community with sufficient volume to get the price down within reason.

The federal government's standards activity continued to wallow along until 1969. By that time the GSA had the world's prime repository of unwanted used computer equipment. Some dated back to the early '60s. The book value of the junk kept them from disposing of it, and the cost of storage soared. The original manufacturers kept raising their maintenance rates until some third party maintenance companies were formed. When this happened, the whole used market opened up. Trading became spirited and competition healthy. This was sparked, in part, by a thorough housecleaning at the BEMA shop in 1968. This resulted in industry assuming the initiative in standards and leading rather than holding back.

While all of this was going on, the users were not idle. One of the big aerospace companies developed a program which would translate from source language to source language, based on tabular descriptions of the two languages. When this was done three things quickly happened: the long sought program interchange suddenly became a reality, the table format required by this program became the language definition standard overnight, and a rash of application-specific languages sprang up. With easy translation, the user could use a convenient language for stating a problem solution and then turn the job over to the central program maintenance group who worked in a consistent single language.

By 1970 several large file-oriented systems had appeared which qualified for the title "data bank." These were large, integrated file systems serving multi-users in both on-line and batch fashions. The groundwork for these efforts had been laid earlier, but until the third-generation hardware and software had settled down, progress was slow, frustrating, and expensive. In 1967, applications analysts realized that basic files had to be built as part of some on-going process. Soon, fully documented "save" tapes started regularly appearing in the archives. Finally, when the tools were available, the files had been stored, and the data bank could be established. The whole thing threatened to flounder until a bright, young programmer teamed with a smart hardware designer to solve the restart problem without extended downtime or unnecessarily expensive backup provisions.

About the same time wisdom and light settled on BEMA, the professional society ice jam melted also. Enough of the founders had retired so that the societies were repopulated by members who considered computing and information processing to be their sole interest and not just a happenstance career. The societies restructured along lines devoted to hardware, software, and applications. Numeric techniques was *one* of the pillars of the applications group. The software society absorbed both SHARE and GUIDE, which had merged in 1968, reorganized along software lines, and discarded their machine orientation, made obsolete by the 360. Each manufacturer still supported a small users group as a sounding board for his products, but the advent of true language standards allowed software to be handled at a level more professional than parochial . . .

(Our observer arrives at the 1973 FJCC exhibits.)

BOB: "Hi, Robert: Moving sidewalks seem to bring old friends back together. How are things in the outerspace industry?"

ROBERT: "Our business volume is holding well. We just got a new trillion dollar contract for another space station. Say, if you're going down to tour the exhibits, let's do the show together."

BOB: "It's good to get out of that crowd. Let's go look at those midget computers mounted in the typewriter stands."

ROBERT: "I'm glad that panel decided these could be considered as a computer, a desk calculator, or as a console depending on how they're used.

"Now, isn't that nice. A computer with 16K memory, add time of 10 microsecs, and modified typer . . . all for $5000."

BOB: "What I like best is the flexibility. The memory bus is brought out to these connectors so we can just plug in this software loader box and preset the software in the Hold-Tite store. That's one way to get the price down: leave all that unnecessary stuff out. Of course, I suppose the price of circuits and core has come down by a factor of 25 in the last 7 years.

"If you get a loader box for each machine it's only $1500 more, but then it can have several software packages and a basic operating system. This will really solve some of our problems back home."

ROBERT: "We have two uses for it right now. That little design group over in the Test Hangar needs a limited amount of computational gear with a customized application language on it. We could set them up for $5K and write the compiler for the big machine. Once we wrote and checked out the compiler at Central we could put it in Hold-Tite and let the Hangar crew work by themselves. It gets rid of that requirement for a secure communications crew to maintain the comm-link from the Hangar to Central.

"The other need I know of is for shipping. We could hook two of them up in tandem to handle the rating and routing function. One typer would have those special forms that have to go with every shipment. The other would be the I/O keyboard for the pair. The clerk can input the load description, measurements, and scale readings. The computer will compute cube, tons, best way, and charges. It can also edit the input and format the output on the slave. If we get stuck and run out of memory we can either buy a loader box for holding rate tables in read-only form or use that acoustic coupler to make the tandem a satellite to the Central."

BOB: "I'd forgotten about that acoustic coupler attachment to the Touch-Tone Fone. With that jack on the side of the new phones, the coupler and the auto-dial can signal over the normal audio phone circuit to Central. You can get it for $100 with no installation charge.

"Let's go find those Immense-Files next."

ROBERT: "Wow! Look at the crowd. I wonder if we'll be able to get close enough to see them work? The advertising brochure claims the Immense-File is to the bulk file what the bulk file was to the drum. I guess that's about right. The Model I Immense-File holds 15 billion bytes in one module and the mechanical part only takes 4 square feet of floor space."

BOB: "I guess the part I like best is the price. It rents for $1500 per month, and the control unit only costs $1000." Let's see. Back in 1966, we could get 800 million bytes on-line for about $6K per month. Now, if we install two Model I's on a single control unit we have 30 billion bytes on-line for only $4000 per month. Say,

---

*IBM Form A22-6843

that's enough capacity at a small enough price to put that engineering technical library on-line, isn't it?"
ROBERT: "Yes, that will easily fit. Those reports average only 50 pages per document, and there aren't but 10,000 of them. If it were all text, that would use up about a fifth of the capacity of two modules, but since we've got to digitize all those graphs and pictures, it probably would take about a half of the twin modules. That would even leave us 15 billion bytes to hold some of that test data they keep crying about because it takes us too long to find the tapes in the archives and get them hung. The test engineer gets tired of sitting at his console before we get the data displayed for him.

"With our new publications system, we can build those technical document files as a by-product of the publication process. In about three years we'll have the most interesting documents stored in the single Immense. We'll probably have to go back and pick up about 10% of those old reports to complete the file; some of that stuff never seems to get too old to be of interest."
BOB: "I wish we could get close enough to see the removable cartridge. I'd like to see what handling problems they've built for us to solve this time. Since this crowd doesn't seem to be getting any smaller, let's move on and come back later. Maybe we can get a private showing since we think we've got a legitimate application."
ROBERT: "Hey—there's a booth over there with absolutely no customers. The contrast is amazing."
BOB: "Oh yes, that's the Bell Tel booth. I think that's made every show for the last ten years in the same form. That big map of the United States with the lightning flashes shooting between those telephone handsets leaves me cold."
ROBERT: "The only thing that's changed since we first saw them back in the '60s is the lads giving the sales pitch. If you change them often enough, they can maintain their sincerity since they don't know they're falling behind two years every year."
BOB: "You'd have thought after that fierce fight over the touch-tone modification that the Bell System would have loosened up a little bit and recognized that the computer was here to stay. It's amazing how the chief engineer fought against modifying the touch-tone handset so that an inexpensive acoustic coupler could just be plugged into a jack."
ROBERT: "Yes, that made a big difference, so I guess all the fight was worth it. I'd like to see the charters for public monopolies changed somehow so the design laboratories of the major manufacturers could cooperate with the big utilities to get a balanced design.

"Do we need to go look at those new graphic terminals over there, or can we see enough from here?"
BOB: "I think I know as much about how they work as I need to know. They haven't changed that keyboard/CRT/hard-copy-printer unit for several years now. I guess the main change in the last year is getting the price down to $5K purchase."
ROBERT: "We installed one of those earlier terminals, and by the time we got through adding the light pen, the scope, the plotter, and all of the special electronics it took, we had almost $6,000 a month going out for a single station. That was a little steep just to give a few engineers the opportunity to fool with graphic displays.

"We ended up scheduling it so tight that the engineer could hardly afford to think at the machine. Fortunately, these new ones will rent for about $120 a month installed and all they need is a pair of twisted copper wires. That makes installation a lot less than it used to be when we had to put in multiconductor coaxial.

"I think the price on these things is finally coming down to a point where the normal engineer can get access to one whenever he wants to without too much trouble, and can keep it as long as he needs it. With these prices, we can get graphical ability out to the design areas where it's needed, and still support them centrally with our multi-CPU. I'm really looking forward to getting some of those Immense-Files on-line so we can hold enough drawings in detail to make up a complete assembly. That should make a big difference in the way we handle engineering changes."
BOB: "Did you see those new optical page readers that can scan a variety of documents and fonts and transmit the data to Central over a twisted pair? They have a programmable digital computer in them just about like the one we saw in those reactive consoles a while ago. The font recognition and document format are all completely programmable—based on tables stored in core. The output is that same acoustic coupler to the touch-tone that made the satellite consoles so popular.

"With the monthly rental getting down to about $200 we may finally be able to collect our primary input data directly from the fundamental documents and eliminate the keyboarding step that's always cost so much. If we put three of them in shipping and receiving, a pair in purchasing, and back that set with one of those new Quik-Disc files, we'll be able to get purchased item status reflected in the new project control system and the financial liabilities report in a way that means something. The price and flexibility of these new devices has made it very worthwhile to consider picking up some of these applications we've talked about for 10 years, but haven't been able to deliver economically."
ROBERT: "That paper on the new master operating system this morning was interesting. I was glad to hear a user instead of the usual manufacturer. Users are always more candid than salesmen. Evidently he's finally getting good operator information so he can figure out what's going on while the machine is processing several jobs simultaneously. As a by-product of this he's getting decent accounting inputs for billing purposes. He said he had trouble getting the system installed initially, but that the thorough set of audit trails they built into the system allowed the troubles to be isolated and corrected easily."
BOB: "I was glad to hear that the fail-soft system qualities were really delivered as promised. We've been living on borrowed time for several years: with our file storage getting bigger and bigger, and those multi-user data banks growing by 10% per year. I wince every time I think of what a catastrophic failure costs us. I still remember that time when we had 200 consoles operating and lost the master index tables to the work files. One thing led to another, and it took us 63 hours to get back on line.

"With this new system working as well as he said it did, I think we could go ahead and order those Immense-Files . . . I think the days of the catastrophic failure are now behind us."
ROBERT: "The multi-CPU shared-storage hardware allowed us to have systems which could reconfigure themselves dynamically in the event of a malfunction. Technically, we owe a lot to the early work done on BUIC and the old FAA system. They finally showed us how to build hardware-software systems which would really fail in stages and not in one big heap.
BOB: "Yes. Those same examples helped us design commercial hardware that reconfigures very nicely in case of a component outage. If we can believe that speaker, the problem can now be solved by changing a few tables and selecting the right pre-stored module. Maybe we're ready to think about going on line with that second batch of 500 consoles now.

"I liked that other paper this morning surveying hard-

ware architecture features. It seems that every major manufacturer now carries some form of compatibility as a fundamental design objective: the 32-bit, register-oriented, variable instruction length machines, based on the 8—bit byte.

"If I understood him, some of them have true hardware compatibility but others have found it sufficient to build a machine which is logically similar so that programs devoid of I/O can be automatically translated from one to the other. The panel discussion left me kind of cold though. I don't really see that the definitions of 'compatible' and 'logical subset' were worth all that time and effort."

ROBERT: "Well, even though they couldn't agree, I know what it means to me. If I have to put any more than 10% of the initial effort into moving from machine to machine, don't bother to send the salesman by to tell me about it. And even if the instructions are completely identical in format and function, if it won't take my file organization or if they fool with the sort order again, just forget it.

"We're already up to the point where most of the cost is in the I/O devices and files. If they gave me a CPU, they couldn't cut my rental more than 5%. I'd just as soon they concentrated their efforts elsewhere."

BOB: "Well, it looks like they've done quite a bit with this new 'triple-vote' modular design. They'll guarantee 95% up-time on all that electronics even if you have only one module. With the new design you can put in the redundant unit and cut the mean time to repair down to 8 minutes. But if you really need the reliability, you can go for that three-processor option with the special control and they'll give you that 'Lloyds of London' policy that will reimburse you for the indirect losses you take based on less than 100% reliability. Boy, that must be reliable stuff.

"We might want to consider putting in one triple-vote system to handle communications, switching, and the message queuing if we decide to get those Immense-Files and put that second 500 consoles on line. With a thousand people on line, we can show a hard saving of $10,000 an hour just in people-time waiting for us to get the system back on line. And this doesn't include anything for the graphics, processing, and data we might have to re-enter or regenerate in case we collapse completely."

ROBERT: "If we go to that second 500 consoles, we'll surely have to pick up the on-line engineering change system, and that carries with it the responsibility of running the manufacturing plant. Management would probably be delighted to have the triple-vote system just to get that Lloyd's insurance covering us against possible business loss.

"The new modules have simplified building control systems out of these general purpose modules. Depending on the box it's in, one of these memory-logic modules can be a part of that souped up console, reduce the cost of those intelligent graphic terminals, or control the scan in that optical page reader. The same unit would also support those on-line medical experiments we're doing in the lab because it interfaces so nicely with analog sensors and digital control devices. They really did well when they got 16K core and 10 microsec add time in a hermetically sealed super flat pack the size of a 3-ring notebook. The power required is so low we can run it off those little thermo-electric sources or the rechargeable cells that have that optional solar face on them."

BOB: "Should we go to that session tomorrow on the user's report describing his experience with the new software? It sounds very interesting."

ROBERT: "I read the preprint and I'd like to attend. He thinks they've finally figured out how to build modular software which is subsettable based on your needs and machine size, but has almost the performance of custom written code. The paper didn't describe exactly how they do it. I hope they explain it in the talk."

BOB: "Didn't the announcement say that part of the modularity lets you routinely change the orientation of the system from real-time to teleprocessing to batch in any combination and with any emphasis?"

ROBERT: "Yes, that's what they tried to tell us. This user maintains they did it and both kept the performance up and the program size down. Furthermore, they have a new language definition for action languages.

"Remember back in the mid-'60s—some of the more theoretically inclined managed to state and describe procedural languages in terms of grammar and syntax? Well, some similar work has just been completed on action languages. They've put some of that same theory to work and come up with a definition of action languages to handle control of the computing system, communications with the operator, all the I/O device actions and scheduling, and the sequencing of processing we used to call job control. It now has some harmony to it, which we've lacked for several years. Matter of fact, I hear it's even easy to understand and has only a few exceptions."

BOB: "Well, if they've managed to do that and also include all those data base handling facilities we've heard so much about, it really ought to be a fine new system. I'm looking forward to the panel discussion after the user talk because I want to ask some questions about these new contract terms.

"Sam Dollar of the Executive Finance Committee heard we were thinking of switching to the fourth generation and sent for me the other day. He pulled out the file with the financial audit of our conversion to third-generation. He had those conversion costs recorded to the penny. The true costs of the conversion were posted to one account and he had two other accounts showing the costs due to the late hardware and the costs due to the late, sloppy software. He made it very clear to me that he wasn't about to approve any expenditures for fourth-generation equipment which didn't include contractual guarantees on the part of the manufacturer. He told me I could order whenever I wanted, but when he signed the contract it better have firm dates for hardware and software with enumerations of required functions and stipulations covering minimum acceptable performance.

"Somewhere he heard we'd been simulating how our proposed workload would pan out in fourth-generation equipment. He advised me—if I couldn't think of a better way—to include that simulated performance as part of the contract.

"I'm going to lay that contracting question in front of tomorrow morning's panel if I have a chance."

ROBERT: "Let's go get a drink. What do you think we should have done in the late 60's to have been better prepared for what we saw today?"

BOB: "Well—let's see. We should have wired that new plant so we could put in a private digital communications network independent of the phone company; placed some restrictions on applications programmers so they wouldn't aggravate the conversion problem by using all of the features of the programming languages just because they were there, kept the descriptions for all those old files we stored so we could figure out the tapes if we ever had to re-read them. I wish I had attended that seminar on assigning functions in multi-computer systems. We should have hired that young guy who knew so much about file organization and integrity. There are several more, aren't there?"

ROBERT: "Yes, I have a few more; but first, where was that DATAMATION cocktail party?" ■

THE NEXT GENERATION

from 50 viewpoints

In an effort to find out what some experienced computer people think may happen with a fourth generation of equipment, the DATAMATION staff prepared some questions and solicited answers. We learned, first, that the people we questioned are surprisingly patient and thorough in taking the time to conjure up ideas about such an ambiguous subject. The questions were divided into three sections, covering hardware, software, and some users' viewpoints.

the hardware

First, let's try to get a handle on the fourth-generation computer. How will you know one when you bump into it? If a consensus of our respondents is any indication—and it should be—the one hardware technology that warrants the fourth generation billing is large scale integration (LSI). (They're quick to add, however, that price per logic function will be a principal factor in this determination.) "This technology (LSI) will have a far greater impact on computer architecture and software than did integrated circuits in the third generation," says a member of a semiconductor house.

While not yet willing to grant that LSI will lead to a computer on a chip, our correspondents variously describe the future of this young upstart as "fantastic" and "extensive." Also "good, measurable, but not revolutionary." This statement, though, is contradicted by still another, who says . . . "No doubt but that LSI will revolutionize the computer field—not only in terms of cost, reliability and size, but also the complexity that becomes reasonable to achieve and the applications that can economically be put on computers. However, this impact will be limited until software and I/O problems are solved, and this may take a long time."

Perhaps some of this will be taken care of by what Ascher Opler in this issue calls "firmware." We

asked to what extent software will become imbedded in special hardware. As expected, the answers ranged from "very little" to "very much." The idea is appealing, they say, but the technology to achieve it economically is doubtful. Among the optimists here, the anticipated firmware functions include subroutines, in-line diagnostics, and I/O systems, but not operating systems and compilers. "This will depend upon how much true standardization evolves."

What about the top circuit speed in Generation Four? Forecasts ranged from 10 nanoseconds to 500 picoseconds, but most estimates were for a propagation delay per gate of 0.5 to 1 nsec. "I do not foresee major increases in top circuit speed, but optical processing might contradict this forecast," said one. The sub-nanosecond speeds predicted here compare with the 5- to 20-nsec per logic level being achieved in today's hardware.

Today's memory hierarchy—only speeded up—may still serve in the next generation; the use of thin films is referenced by only a few, the associative memory was mentioned by only one, and "biological devices for the intermediate level" got a vote. The most clearly delineated hierarchy: "registers and multiple or general registers implemented with integrated circuits; IC scratchpad memories (50-200 nsec); main memories implemented with plated wires, thin films and, for a while, cores (100-500 nsec); on-line auxiliary storage, largely discs and drums but also some solid-state random access devices (e.g., plated wires or etched permalloy) and BORAM devices (particularly for military and later commercial applications); and off-line tapes, discpacks, magnetic cards, etc."

The CPU organization may be in for a radical departure, if a slim (58%) majority of our counsellors has its say. They turned right around, however, and voted down the idea that parallel processor organization, such as Solomon, might come into use. One said parallel processors, yes, but not like Solomon. Although not ordinarily considered "organization," the idea that multiprocessing would come into its own in the fourth generation received a definite "yes" vote. Implied in this is the growing knowledge that we still don't know how to multiprocess, but we will soon.

How soon? When will you get your first chance to collide with a fourth-generation mainframe? Fifty percent said between 1971 and 1975: 25% said as early as the end of 1970: the rest didn't know. Among those who predicted the earlier date—by the end of 1970—was the one designer who said he'd know one when he saw one because its new technology would be cryogenics.

software

In asking our patient respondents about software for the hypothetical fourth-generation equipment, we made certain (doubtless debatable) assumptions about the nature of past and present programming. It was assumed that first-generation software included machine-language programs, subroutines, and assemblers. The second generation added higher-level languages, monitors, and macro-assemblers. The third generation brought operating systems, conversational time-sharing, multi-programming, and data management systems.

Considering these developments as characteristic, we asked, what will the fourth generation bring?

One of the most interesting things about the responses was the discovery that there is a clear consensus of opinion about some aspects of software wanted and flaming disagreement about others.

The first question asked if the respondents would "expect" fourth-generation computers to be accompanied by a radical departure in software. There was about an equal split of yes and no answers, with two people noting that they "hope so" but don't really expect it. A few others pointed out that software development tends to be gradual and is not likely to be closely tied to a new generation of equipment. In fact, "fourth-generation hardware is likely to be adolescent when fourth-generation software is born." There are some feeling that the user might have software allowing him to define his own language, while another said that "we'll be lucky to have truly second-generation software with our fourth-generation hardware."

Another virtually even split between yes and no answers resulted from a question asking if paging techniques will come into common use. But some said "cheaper mass stores will win."

A question asking if English-language programming would become a major factor drew a variety of interesting comments, although a majority voted either "no," "it's doubtful," or "I hope not." Several respondents saw it as a growing and significant factor in certain applications, such as information retrieval. And one of those saying yes put it this way: "It must if the computer is to become an every-day, commonplace business and household machine—and I'm sure the manufacturers want to see it become just that." One of the best answers on the negative side: "No. Natural language compiling is unnatural."

There was general agreement among those surveyed that nonprocedural languages would be of growing importance, although some felt that progress would be very slow. One comment: "Nonprocedural languages have been a sought-after convenience since the early fifties . . . Everyone was and still is looking for that software package which allows you to say 'Do payroll' . . . the user would describe only the problem parameters in terms of environmental conditions to the software systems which in turn would grind out a maximum solution to this particular problem. One could wonder whether the parameters concerned with supplying the problem description would not be as overpowering and burdensome as the actual programming requirement is."

Our respondents also had some vigorous comments on what methods will be used to maximize processing efficiency in an information utility using fourth-generation equipment—including the opinion that there won't be any information utilities doing processing; they will just serve as data banks. Some think that the best method would be fragmentation of applications into special-purpose, dedicated utilities—such as credit information, legal information, etc. But several answers reflected the viewpoint that rising computation speeds and shrinking cost would make processing efficiency a minor problem, compared to the present.

How should information security be handled in an information utility? "Very carefully" is a fair summary of the answers. Some methods suggested: a pattern-recognition device that can recognize fingerprints; a true book code, or private language, instead of a cipher; time-dependent codes; remote data storage at the user's location; individual scrambler/descrambler keys; voice recognition.

Suggestions were requested for types of planning that would simplify future conversion efforts. Typical recommendations were thorough documentation ("overdocument"), use of

higher-level languages, and continuity of languages by the manufacturers. There is also some feeling that users can't stand another conversion—that further changes must be evolutionary, considering the massive size of some applications programs even now.

A related question was also asked: "Will there be an 'OS 360 backlash' that will lead users to insist on less complicated software?" Although one respondent said the question contains a false implication, is loaded, and is unanswerable, it provoked longer and more lively answers than any of the other questions. A large majority think there is and will continue to be a demand for simpler software—but many add that it should be simpler only in terms of user convenience, not function. Several people foresee more customized systems coming, closely tailored to fit a user's needs. Some of the more bitter comments: "By the time the . . . backlash . . . gains any momentum, the large-scale machine users will have implemented their own operating systems" . . . "OS is completely compatible; it makes the Mod 75 run like a Mod 30" . . . Some think, however, that the problems illustrated by OS 360 can be overcome, given better check-out, training manuals, and general customer support.

There is strong agreement that software for a new generation of equipment should not be confined to a single, standard language—unless it were some sort of macro-language that allowed users to develop a special dialect fitting their own needs. An accompanying question, asking about critical bottlenecks with present software, resulted in such descriptive phrases as "trying to satisfy everyone . . . compromises between conflicting goals . . . attachment to 1957-vintage compiler . . . ponderousness."

We also asked if manufacturers will be selling software separately from hardware. There was no clear-cut consensus in the answers, but a majority think this is likely to come about—restricted to special cases. Many mentioned the recent announcement by Scientific Data Systems that COBOL will be sold separately as evidence that the trend is starting. A few said that the federal government will force the issue. Other responses: it's more likely to be done by separate companies. It was also suggested that separation will come about only if the patent problem is resolved. One other idea: design the machines to keep track of software use automatically and charge for it; then the price would be separated from the hardware cost no matter who wrote the software.

Our last question: "What sort of software would you like to see?" A brief summary of the replies: smaller conceptual jumps, standard format and documentation, more debugging aids, easier to use and maintain, simple and conversational.

Besides offering some ideas for the future, perhaps the variety of answers to our questions about what the computer community wants will give readers some sympathy for the problems of the manufacturers.

some users' views

Then there's the ever present user, tormentor and tormented, who must struggle through the conversion to each new group of machines—sprouting greys over things like down-time, complex operating systems (and delays), new compilers and delays and programmer education, acceptance tests, core memories wired in backwards, the wrong color on his tape unit, and a boss who wonders why they didn't buy the all purpose 720/140 just announced which sounds like tomorrow but won't be here until later, if then.

This is not to indicate more than that each change of equipment has its attendant problems. A survey of veteran computer users shows they felt the current generation of equipment brought a hoard of advances over the previous one: improved cost/performance; larger, high-speed main memories; higher capacity, lower cost random access devices; less down-time; wider choice of peripherals; greater multiprogramming, multiprocessing capability; expandability and compatibility of computers in a family, more sophistication at a lower mainframe level (i.e., multiprogramming, teleprocessing, on-line terminals, graphics, etc.); increased use of operating systems in OLRT environment; development in file management systems; program compatibility at assembly language level; emulation ("in our experience, the first really complete preservation of programming"); and who said "greater IBM profits"?

And almost to the item, the next generation, due by consensus between 1971-75, should have improvements on these improvements. Users felt that what essentially will shove a system into a new era is a mainframe that increases speed by an order of magnitude over what's now available, increases internal mass storage to the billion-character range, and decreases cost (thin film and cryogenic core are offered). Other hardware wants are larger, more reliable, cheaper random access storage, although this is seen by one user as less important if internal memory is increased and improved; larger varieties of special purpose remote devices, including 3-D and color displays, handwriting and voice input terminals. Although many felt some peripherals should be speeded up by two-four times (5,000 lpm electronic printer, 600 KC discs), the emphasis was on lower cost, greater reliability, and, for the printer, increased capability (character sets, multifonts) and quality. High-speed, low-cost communications, definitely a lack currently, will be vital for the activities of any coming generations. One user called for low-cost high-bandwidth microwave links to centralized facilities for more powerful time-sharing systems. Better communications interfaces are also required.

No, software is not being avoided. It was the most popular answer to the question on critical problems still evident in third generation systems. The tally really isn't in from the field yet on this generation, but users reflected the disgruntlement with the delays, the debugging and education problems, inadequacy of manuals, the cost of conversion. Some answers for the next machines were offered, such as more software functions being built into hardware, less operating system for the average user (some feel they have a lot more computer than they can figure out how to use). Definite wants are for fourth generation compatibility with the predecessor family, and for separation of hardware and software pricing "so we'll know what we're paying for." Half surveyed thought this separation was really coming.

The idea of a universal language was appealing to some, simply because conversion problems would phase out and a vast pool of programs would become available to users of differing systems. A stable basic source language for each system seemed more practical, and most users are content to settle on improving what exists, rather than "compounding the confusion."

The last dampening word on software was offered by a dp director at a major insurance firm, "Assuming problems of reliable operation of hardware are solved, there is little real promise of simplification of program writing and system construction. This may be fine from the point of view of more jobs for people, but it's not so fine from the businessman's view of lower cost operations." ■

DATAMATION®

35 MASON ST.
GREENWICH, CONN. 06830

SALES OFFICES

35 Mason St. Greenwich, Conn. 06830 203 661-5400	112 West Haven Road Manchester, N.H. 630 NA 5-9498
205 W. Wacker Dr. Chicago, Ill. 60606 312 FI 6-1026	1830 W. Olympic Blvd. Los Angeles, Calif. 90006 213 385-0474